Computer Science

INVESTIGATION OF SAT-BASED SCHEDULING TECHNIQUES IN COMPARISON WITH TRADITIONAL APPROACHES, Ricky P. Guidry, Stefan Andrei*, Lamar University, Department of Computer Science, Beaumont, TX, 77710, sandrei@cs.lamar.edu

Given a set of tasks, the scheduling problem means to find a particular schedule such that all the tasks are executed by a processor before their time deadlines. In our modern society, the scheduling problem has significant applications in many technological areas, such as, real-time embedded systems, and networks. Since the scheduling problem is hard, researchers look for efficient heuristics to solving it. Similar to preparing a schedule of tasks that must be done in everyday life, scheduling a set of computer tasks, or processes, means to determine when to execute which task, thus determining the execution order of these tasks. A periodic task has many instances, and there exists a fixed period between two consecutive releases of the same task. In case of a multiprocessor or distributed system, scheduling implies also determining which processors get each specific task.

A task T is characterized by: S – the start time, or release time; c – computation time, also considered as the worst case execution time; d – the deadline; p – the period. There usually exists in practice additional constraints that may complicate scheduling the tasks within the deadlines. For example, dealing with task that share resources, or whether the task can be preempted. Many of the practical scheduling problems are known to be Nondeterministic Polynomial-Time Hard (NP-hard). An informal definition of NP-hard is simply that the problem is "at least as hard as the hardest problems in NP". The difficulty of NP problems is that the computation time is directly proportional to the size of the problem. In simpler terms the harder and more complex the problem the more computation time is required to calculate the solution.

The most popular scheduling heuristic techniques are *rate-monotonic* (RM), *earliest-deadline-first* (EDF), *least-laxity-first* (LLF). All these techniques offer solutions in polynomial time for large classes of tasks sets. An algorithm is said to run in polynomial time if the run time is no greater than a polynomial function of the problem input size. A set of tasks are considered to be feasible if there exists a scheduler can schedule all the tasks. In other words, a set of tasks is feasible if each task can complete execution by its deadline.

Many researchers have been working on the Satisfiability (SAT) problem for a long time and many researchers have successfully contributed to the field. The SAT problem is determining whether the propositional formula can be equated to true and is one of the most central problems in computer science. The purpose of our research is to investigate how to transform a non-feasible task set into a feasible task set, while minimizing the deadline increase. Our work investigates which task's deadline has to be increased to make the set of tasks feasible. The novelty of this work is that our algorithm is based on the SAT problem. Using the C++ programming language we have implemented a program that will take a task set as input, encoded the task set into the DIMACs format and output the encoded task set. The encoded task set is then run through the zChaff SAT solver. If the encoded task set is found to not be feasible then the deadlines of the original task set are increased slightly. Then the task set is run through the program again. This process is repeated until the unfeasible task set is found to be feasible.

Experiments on the implementation of this project have been conducted and the results are promising. We used an Intel Pentium3.20 GHz processor, using 2 GB of memory.

Ricky Paul Guidry was supported by LU-REG grant of Dr. Stefan Andrei